COSC 2231- Programming 1

Tutorial 1 - Basic Programming concepts

Review Questions:

1. What does the term "source code" refer to?
2. What is a syntax error in a java program?
3. What is a runtime error in a java program?
4. What is a logic error in a java program?
5. What is an "identifier" in a java program?
6. What are the rules governing what a valid identifier can contain?
7. What is a String literal?
8. What are the two ways you can define a String (literal)?

Tutorial Exercises:

1. **What do the following steps in the program development lifecycle involve:**

   a. **Editing**

   b. **Saving**

   c. **Compiling**

   d. **Executing**

   e. **Debugging**

2. **How does the compilation of java programs differ from programs written in other languages?**

3. **How do you run a java program and what is required to run it?**

4. **The** `main()` **method needs to be preceded by the following three words:**

   _____ _____ _____

5. **Explain which of the following identifiers are *invalid* and discuss why they are incorrect.**

```
a)   A1              b)   1A         c)   $2_coins       d)   Do_it

e)   fish&chips      f)   got___it   g)   _zen           h)   z00m

i)   100_cents       j)   stop!      k)   Two numbers    l)   Seven-11

m)   totalNumberOfStudentsEnrolledInThisTutorial    n) totStdsInTut
```

6. **Answer the following questions in relation to the program below:**

```
public class HelloWorld
{
   public static void main(String[] args)
   {
      System.out.print("Welcome ");
      System.out.println("To ");
      System.out.print("RMIT");
   }
}
```

a. Is the declaration of the class "HelloWorld" necessary?

b. What role does the main method play in the program?

c. What is the program doing in the first line of the main method?

d. What is the difference between System.out.print() and System.out.println()

e. What results does this program produce?

7. **The following program has some errors in it. List the errors and state what is wrong with them.**

```
//  /*  programs need
        comments  */

class OneOne
{

   public static void main(String[] args)
   {

      aString = new String("Well!");
      int  2ndString = "Now!";

      System.out.println("Isn't this an interesting program);
      System.out.println("Answer yes or no!")
   }
}
```

COSC 2231 - Programming 1

Tutorial 2 – Variables and operators

Review Questions:

1. What is a variable used for?
2. What is a primitive type and what primitive types are available in java?
3. How do you define a constant value in java?
4. How do you set up a BufferedReader object to read input from the keyboard?
5. How do you set up a Scanner object to read input from the keyboard?
6. What is the order of precedence for arithmetic operators in java?
7. How does the modulus (remainder) operator work in java?
8. What are the three commonly used unary operators in java?

Tutorial Exercises:

1. **Give appropriate declarations for variables used to store the following values:**
   - **a.** An employee's salary
   - **b.** The month number within a year
   - **c.** An employee identification number
   - **d.** The constant string "Hello"
   - **e.** The capacity of a tank in cubic inches
   - **f.** Number of days passed since 1/1/2000
   - **g.** A letter to hold an answer ('y' or 'n')
   - **h.** The number of assignments in PP1
   - **i.** The average mark for all PP1 students
   - **j.** The conversion factor from $AUS to $US

2. **What is the output of the following statements:**
   - **a.** `System.out.println(10.0 / 3.0 + 5.0 * 2.0);`
   - **b.** `System.out.println(10 – 3 % 5 * 2);`
   - **c.** `System.out.println(10 / 3 + 5 / 2);`
   - **d.** `System.out.println(13 % 5 / 2);`
   - **e.** `System.out.println((10 + 3 / 2) * 3);`
   - **f.** `System.out.println(5.0 % 3.0);`

3. **What is the value of a in the following (assume a is int)?**
   - **a.** `a = 45 / 8 * 4 + 2;`
   - **b.** `a = 17 + (21 % 6) * 2;`
   - **c.** `a = (4 * 2 + 2) * 2;`

   Recalculate the results for expressions b and c above – are either of these expressions evaluated differently without the brackets.

4. **What is wrong with the following expressions, one trying to compute the area of a semi-circle and the other trying to compute the positive root of a quadratic equation? How do we fix these problems?**

   a. `areaSemiCircle  = 1/2 * Math.PI * r * r;`

   b. `root1 = (- b + Math.sqrt(b*b – 4*a*c)) / 2 * a;`

5. **Which of the following expressions are equivalent to the statements below:**

```
y = x;
x = x+1;
```

a. `y = x++;` **b.** `x = y++;` **c.** `y = ++x;` **d.** `x = ++y;`

6. **Let** `y` **have the value** `5` **and** `z` **have the value** `8`**.**
   What are the values of `x`, `y` **and** `z` **after each line of the following fragment?**

   a. `x = y++ + z++;`

   b. `x = y++ + ++z;`

   c. `x = ++y + z++;`

   d. `x = ++y + ++z;`

7. **What will be the output of the program below given the input below?**

```
import java.util.Scanner;
public class ScannerDemo2 {
   public static void main (String[] args)
   {
      Scanner keyboard = new Scanner(System.in);
      double n1 = keyboard.nextDouble();
      double n2 = keyboard.nextDouble();
      String message = keyboard.nextLine();
      System.out.printf("n1 = %8.2f n2 = %8.2f message = %10s",
                        n1, n2, message);
   }
}
```

```
12.5
13.5
This is the end
```

← Input one item per line

How can you modify the program to print "`This is the end`**" for message?**

Tutorial 3 – Variables, operators and comparisons

Review Questions:

1. How much space is allocated for each of the primitive numeric types in memory?
2. How are expressions that contain different types of values evaluated?
3. What are the rules for governing the assignment of values to variables in java?
4. What are the commonly used equality operators in java?
5. What is an object type made up of? Give an example of an object type.
6. How do you compare object type "values" in java.
7. What are the commonly used relational operators in java?
8. What is the concatenation operator and how does it work?

Tutorial Exercises:

1. **Using a "trace" table of the form :**

	Current Value
variable 1	
variable 2	
… …	

"trace" the execution of the following code segment (i.e. write down each value assigned to each variable).

```
final double MYPI = 3.14;
final int ONE = 1;
final char BLANK = ' ';
final boolean AFFIRMATIVE = true;
int i1, i2;
boolean b1, b2;
char c1, c2;
float r1 = 2, r2 = 0;

i1 = ONE;
i2 = i1 + 2;
r1 = (float) (i1 / i2);
i1 = i2 - i1;
r2 = r2 + (float) ONE;
c1 = 'A';
c2 = BLANK;
b1 = false;
b2 = c1 == c2;
b1 = MYPI > (float) ONE;
b2 = AFFIRMATIVE && b2;
```

2. **Given the types of the variables** `x`, `y` **and** `z` **are** `int`, `float` **and** `double` **respectively, which of the following statement(s) are likely to result in error ?**

a) `x = y;` **b)** `x = z;` **c)** `y = x;` **d)** `y = z;` **e)** `z = x;` **f)** `z = y;`

3. **The following code segment prints an incorrect result (instead of the expected** `sqr = 10000000000` **). What is the problem and how can we correct it?.**

```
int num1 = 100000;
int sqr = num1 * num1;
System.out.println("sqr = " + sqr);
```

4. **For each expression on the left column find the matching expression on the right column:**

1. (X < Y) && (Y < Z)	**a.** !(X != Y) && (Y = Z)
2. (X > Y) && (Y >= Z)	**b.** !((X <= Y) || (Y < Z))
3. (X != Y) || (Y = Z)	**c.** (Y < Z) || (Y = Z) || (X = Y)
4. (X = Y) || (Y <= Z)	**d.** !(X >= Y) && !(Y >= Z)
5. (X = Y) && (Y = Z)	**e.** !((X = Y) && (Y != Z))

5. **What is the value returned by the following expressions. Explain carefully.**

   a. `"Mary".compareTo("Mary")`
   b. `"Mary".compareTo("Matt")`
   c. `"Pam".compareTo("pam")`
   d. `"mary".compareTo("Mary")`
   e. `"  mary".compareTo("mary")`
   f. `"mary".compareTo("mary anne")`

6. **What output is produced by the code segment below? How the code segment can be modified to swap them correctly.**

```
int x = 5, y = 8;
// attempting to swap the values
x = y;
y = x;
System.out.println("After swapping: x is " + x + " and y is " + y);
```

7. **What will be the output of the program below ?**

```
public class Compare
{
   public static void main (String[] args)
   {
      int num1 = 10, num2 = 10;
      Integer rnum1 = new Integer(20);
      Integer rnum2 = new Integer(20);

      if( num1 == num2)
         System.out.println ("num1 is equal to num2");

      if( rnum1 == rnum2)
         System.out.println ("rnum1 is equal to rnum2");
      else
         System.out.println("rnum1 not equal rnum2");

      if( rnum1.intValue() == rnum2.intValue())
         System.out.println("objects have same value");
      System.out.println();
   }
}
```

COSC 2231 - Programming 1

Tutorial 4 – Decisions

Review Questions:

1. What is the basic structure of a decision statement?
2. How are the conditions structured in a decision statement?
3. How do you compare Strings in a decision statement?
4. How do you compare other "object type" values in a decision statement?
5. How can you check for multiple conditions in a decision statement?
6. What effect does nesting decision statements inside each other have?
7. What is a multi-way decision and how are they handled in java?
8. What is the conditional operator and how does it work?

Tutorial Exercises:

1. **Complete the following Java code segment so that the messages "You have passed CS108" and "You may proceed to next level" will be printed (using two separate println statements) if the input marks is greater than or equal to 50. Otherwise the messages "Sorry, you have failed CS108" and "You may re-enroll next semester" should be displayed (using two separate println() statements).**

```
// assume student marks is read and stored in the variable marks
if ( ... )
   ...
```

2. **Write a program to compute the overall result for the subject CC123. The subject has a course hurdle and an exam hurdle, both of which are marked out of 100. A student is considered to have passed the subject if the mark for both each of the hurdles are greater than or equal to 40 AND the overall mark (the average of two hurdles), is greater than or equal to 50. All other cases are considered to be a fail.**

   The formula for the overall mark is: *(course result + exam result) / 2*

   Test your program with the following data. The expected result is shown next to it.

```
course = 45 exam = 65        (expected result is pass)
course = 30 exam = 90        (expected result is fail)
course = 30 exam = 36        (expected result is fail)
course = 60 exam = 90        (expected result is pass)
```

   Start by checking for the "pass" criteria in a single if statement first and handling the "fail" case in an else clause.

3. **Rewrite the decision structure from question 2 by breaking up the condition for a pass result in the if statement so that it becomes three separate decisions and use nested if statements to check for the pass result condition. Is there any advantage to handling the decision making process this way, as opposed to the decision structure in question 2?**

4. **Rewrite the decision structure from question 2 by identifying the different "fail" cases (course, exam and overall) first using a multi-way decision structure (the "pass" case will be handled in the else clause this time). Is there any advantage to handling the decision maing process this way as opposed to the decision structure in questions 1 and 2?**

5. **Construct a truth table for the following decision structures:**

```
if (test_1)
   statement_1;
else if (test_2)
   statement_2;
else
   if (test_3)
      statement_3;
   else
      statement_4;
```

```
if (test_1)
   statement_1;
else
   if (test_2)
      if (test_3)
         statement_3;
      else
         statement_2;
   else
      statement_4;
```

Are the two code segments the same? Explain carefully.

6. **Construct a truth table for the following code segment. Once you have constructed the truth table, derive a complete set of test data to test the code segment. What does the code segment do?**

```
int a, b, c;
int temp;
//  assume a, b and c have been given values
if (a > b)  {
     temp = a;
     a = b;
     b = temp;
}
if (b > c)  {
     temp = c;
     c = b;
     b = temp;
}
if (a > b)  {
     temp = a;
     a = b;
     b = temp;
}
```

7. **Rewrite the following Java code segment so that the same output occurs, but no output statement contains the same word(s) as any other output statement. E.g. the word "integer" should appear in one println() statement rather than all three println() statements.**

```
if (k >= 0)
   if (k <= 9)
      System.out.println ("integer is positive digit") ;
   else
      System.out.println ("integer is positive non-digit") ;
else
   System.out.println ("integer is negative") ;
```

Tutorial 5 – Repetition, Parameters and Arrays

Review Questions:

1. What is the basic structure of the for and while loops?
2. How is a do-while loop structured? How does it differ from the for and while loops?
3. What is a nested loop and how does it work?
4. What do the break and continue statements do?
5. What is an array and how can it be used to store data?
6. How do loops help with processing data in arrays?
7. What are command line arguments and how do they work?

Tutorial Exercises:

1. **Write for loops / nested for loops to print the following.**

   a) **2 3 ... 10**

   b) **20 30 40 ... 100**

   c) **1 2 4 5 7 8 10 11 ... 98 100** **(numbers from 1 to 100 which are not a multiple of 3)**

   d)
   ```
   1   2   3  ... 10
   2   4   6  ... 20
    .   .   .
   10  20  30 ... 100
   ```

   e)
   ```
   1    2   3 ... 10
   11  12  13 ... 20
   .   .   .
   91  92  93 ... 100
   ```

   f) **a b c … z**

   g) **A B C … Z**

2. **Trace the execution of the following code and determine its output.**

```
final int size = 5;
   for (int row = 1; row <= size; row++)  {
      System.out.println();
      for (int col = 1; col <= size; col++)
         System.out.println(row);
   }
```

3. **Write code segments to produce the following output**

a)
```
12345
1234
123
12
1
```

b)
```
5
54
543
5432
54321
```

c)
```
*
**
***
****
*****
```

d)
```
    *
   ***
  *****
 *******
*********
```

4. **Print the series 10 20 40 80 160 320 … until the sum of the series first exceed 100000.**

5. **Write a do loop which will continue to read positive integers and sum them until two consecutive integers entered are identical. The second identical integer should not be included in the sum. For example, if the user input 10 20 10 30 30 then the final value of sum should be 70 (10 + 20 + 10 + 30).**

6. **What will the output of the program below ?**

```
public class TestArray{

   public static void main (String[] args){

      int nums[] = { 12, 25, 20, 40, 10};
      int sum = 0;

      for (int i=0; i<5; i++)
         sum += nums[i];
      System.out.println(" A. "  + sum);

      sum = 0;
      for (int i=0; i<5; i=i+2)
         sum += nums[i];
      System.out.println(" B. "  + sum);

      sum = 0;
      for (int i=0; i<5; i++)
         if ( nums[i] < 30)
            sum += nums[i];

      System.out.println(" C. "  + sum);
   }
}
```

output A. ____ output B.____ output C.____

Write a for loop to print the array elements in reverse order (i.e. 10, 40 … 12)

7. **What will be the output of the program below ?**

```
public class TestArray2{
   public static void main (String[] args){

    int nums1[] = { 10, 25, 20, 40, 10};
    int nums2[] = new int[5];

    for (int i=0; i<5; i++)
       nums2[i] = nums1[4-i];

    for (int i=0; i<5; i++)
       System.out.println(nums2[i]);

   }
}
```

8. **What will be the output of the program below ?**

```
public class TestArray3{
   public static void main (String[] args)
   {
     int nums1[] = { 10, 25, 20, 40, 10};
     int nums2[] = { 30, 50, 67 };
     // size of new int array = size of 1st arry + size of 2nd array
     int nums3[] = new int[nums1.length + nums2.length];

     for (int i=0 ; i< nums1.length ; i++)
        nums3[i] = nums1[i];

     int l = nums1.length;  // length of first array
     for (int i=0; i<nums2.length; i++)
        nums3[l+i] = nums2[i];  // copying from 2nd arry but with offset l

     for (int i=0 ; i< nums3.length ; i++)
        System.out.println(nums3[i]);
   }
}
```

COSC 2231 - Programming 1

Tutorial 6 – Introduction to designing/writing classes

Review Questions:

1. What is the purpose of a class?
2. What is an object and how is it stored?
3. Where is the data for an object stored?
4. How is the process of setting up or initializing the data in an object handled?
5. How are objects created in a program?
6. How do we access the data stored in an object?
7. How do we change the data stored in an object?

Tutorial Exercises:

1. **Given the Account class and the statement**

   Account temp = new Account("Tom", 1000);

   Write a statement to show:

   (a) how the Account reference temp can be changed
   (b) how the Account object itself may be changed

2. **Suppose we are required to keep track of the telephone numbers of all bank account holders (as a string) in addition to name and balance.**

   What changes will you make to the Account class

   a. **Instance variables?**

   b. **Constructor -**

      ***public Account(String accountID, String accountName, double amount)* ?**

   c. **Accessors?**

   d. **Mutators?**

3. **Redefine the transfer method() in terms of deposit() and withdraw() methods.**

4. **Suppose we are required to provide an operation that allows us to add interest (given the current interest rate) to all account holders.**

   a. **What additional method will you provide ?**

   b. **Give a possible implementation for that method.**

5. **Implement a method to print the details of an Account object to the screen in the following format:**

```
ID: a1234

Name: Bob the Builder

Balance: $1243.87
```

6. **Override the toString() method so that it returns the details of an Account in a format similar to that shown in question 5 above.**

   (Hint: use the concatenation operator to "build" each line of the required output as well as to "join" the lines together to form the one string).

   How would you use this method to display the details for an account object from the main program?

7. **In groups of 3-4 students discuss the design for a class to model a simple Person object.**

   You will need to consider the following details in your discussion:

   a. **What information is common to all types of people?**
   b. **How would you create a new person object in another part of the program?**
   c. **What information might we need to retrieve from a Person object?**
   d. **What information might we need to change in a Person object?**

8. **How might we change the Person class we have designed so that there is a connection between a Person and and the parents of that person (which are also Person objects)?**

   Discuss how the connection between a given Person and the two Person objects which represent the parents of the Person in question could be set up in the main program.

COSC 2231 - Programming 1

Tutorial 7 – Working with arrays of objects

Review Questions:

1. How is an array actually structured?
2. How is an array of "object type" elements structured?
3. How do you create an array of object type references? Can the array elements be used immediately after the array has been created?
4. How do you fill an array of objects with "data"?
5. How do you retrieve an object from an array of objects?
6. How do you search for an particular object in an array of objects?
7. How do you process all of the objects in an array of objects (eg. print the details for each object in the array)?
8. What happens when you manipulate the array elements themselves (rather than the objects they are "storing")?
9. How does parameter passing work for methods?

Tutorial Exercises:

1. **Assume that the application class below makes use of the Account class defined in the lecture.**

```
import java.io.*;
public class TestAccounts2 {
   public static void main(String args[]) throws IOException {
         String accountID, name;
         double balance;
         BufferedReader stdin = new BufferedReader(
                              new InputStreamReader(System.in));
         Account[] a = new Account[3];
         a[0]  = new Account("s5234","David",1000);
         a[1]  = new Account("s1239","Cliff",2000);
         a[2]  = new Account("s4236","Martin",3000);
         //  REPLACE THIS COMMENT WITH ONE OF THE FOLLOWING
         System.out.println("Account Details");
         for (int i=0;i<3; i++)
             System.out.println("ID = " + a[i].getID() +
                                " name = " + a[i].getName() +
                                " balance = " +a[i].getBalance());
   }
}
```

What will be the output if we replace the indicated comment with each of the

following blocks? Explain the output.

a.
```
a[1] = a[0];
a[2] = a[1];
a[0] = a[2];
```

b.
```
a[0] = a[1];
a[0].withdraw(100);
a[1].withdraw(100);
a[0].transfer(a[1],500);
a[1].transfer(a[2],500);
```

c.
```
for (int i=0;i<3; i++)
   if (a[i].getBalance() > 1500)
      a[i].deposit(10);
```

d.
```
Account p,q;
for (int i=0;i<3; i++)
{
if(a[i].getName().compareTo("Martin") == 0)
   p = a[i];
else if(a[i].getName().compareTo("David") == 0)
   q = a[i];
}
p.transfer(q,100);
```

e. **Replace the for loop with a do while loop so that the user may repeatedly print the account details by specifying the account ID. If no such ID exist print appropriate error message. After each iteration the user should be prompted whether to continue.**

```
Enter Account ID:   s4236
Acount details: Name = Martin  Balance = $3000
Continue (Y/N) ? Y

Enter Account ID:   s5234
Acount details: Name = David  Balance = $1000
Continue (Y/N) ? Y

Enter Account ID:   s5444
No such account exist
Continue (Y/N) ? N

Bye for now.
```

(Hint: You need to use the getID() method)

2. **Which of the following is the correct for loop statement to access the first 3 elements of an array a?**

   a. `for (int i=0; i<=3; i++) a[i] = …`

   b. `for (int i=1; i<3; i++) a[i] = …`

c. **`for (int i=1; i<3; i++) a[i] = …`**

d. **`for (int i=0; i<3; i++) a[i] = …`**

3. **What will be the output of the program below ? Why ? Trace through with diagrams.**

```
class MyInt {
   public MyInt(int n) {     val = n;   }
   public void setVal(int n) { val = n;   }
   public int getVal() {      return val;   }
   private int val;
}

public class TrySwap {

   public static void intSwap(int x, int y) {
       int temp = x;
       x = y;
       y = temp;
   }
   public static void refSwap(MyInt x, MyInt y) {
       MyInt temp = x;
       x = y;
       y = x;
   }

   public static void contentSwap(MyInt x, MyInt y) {
       int temp = x.getVal();
       x.setVal(y.getVal());
       y.setVal(temp);
   }

   public static void main (String[] args) {
      int u = 10;
      int v = 20;
      MyInt num1 = new MyInt(10);
      MyInt num2 = new MyInt(20);
      intSwap(u,v);
       System.out.println("u = " + u + " v = " + v );
      refSwap(num1, num2);
      System.out.println("num1 = " + num1.getVal()
                 + " num2 =" + num2.getVal() );
       contentSwap(num1, num2);
      System.out.println("num1 = " + num1.getVal()
                 + " num2 =" + num2.getVal() );
   }
}
```

COSC 2231 - Programming 1

Tutorial 8 – Inheritance and Subclasses

Review Questions:

1. What do the terms superclass and subclass refer to?
2. What is the relationship between a superclass and a subclass?
3. How do you define a subclass?
4. How do you decide what functionality should be implemented in the superclass and what can be left for the subclass to define later on?
5. What do the terms inheritance and accessiblilty refer to?
6. What are visibility modifiers and how do they affect inheritance/accessiblilty?
7. How do you refer to functionality defined in the superclass from within the subclass?
8. What do the terms overriding and overloading refer to?
9. What does composition refer to when designing/writing a class?

Tutorial Exercises:

1. **In the following pair of classes identify the superclass and subclass:**
   (a) Employee, and Manager **(b) Polygon, Triangle** **(c) Vehicle, Car**
   (d) Object, Rectangle **(e) Integer, Number**

2. **If class C extends class B and class B extends class A then which of the following statement is true ?**
   a. **an instance of C can use all public methods of classes A, B and C**
   b. **an instance of A can use all public methods of classes A, B and C**

3. **In the program below a student derived a subclass to model fixed rate savings account. Since the constructor for the subclass takes an additional argument, the (fixed) interest-rate, he added an instance variable intRate. As he was unable to access the superclass private instance variable directly, he added an additional instance variable balance in the subclass. The driver program he wrote to test his class added interest for 3 months before displaying the final balance. The output showed that the interest has not been added.**

   Identify the problem and correct it.

```
import java.io.*;
import java.util.StringTokenizer;
public class SubTest {
   public static void main(String args[])
   {
       FixedRateSAccount account =
              new FixedRateSAccount("f1234",10000.0,1.0);
       account.addInterest();  // intersest for Jan
       account.addInterest();  // intersest for Feb
       account.addInterest();  // intersest for Mar
       account.print();
   }
}
```

```
class Account
{
   public Account(String accountID, double amount)   {
       accID = accountID; balance = amount;
    }
    public void deposit(double amount)    {
       balance = balance + amount;
    }
    public boolean withdraw(double amount) {
        if (balance >= amount)
        {
            balance = balance - amount;
            return true;
        }
        else return false;
    }
    double getBalance() {
        return balance;
    }
    void print() {
          System.out.println("ID = " + accID + " Bal = " + balance);
    }
    private String accID;
    private double balance;
}
class FixedRateSAccount extends Account
{
    public FixedRateSAccount(String accountID, double amount,
                                                double intRate)
        super(accountID,amount);
        this.intRate = intRate;
    }
    public void addInterest()
    {
       balance += balance*intRate/100;
    }
    private double balance; // same as superclass
    private double intRate;
}
```

4. **Extend the account class to create a new subclass called CAccount for modeling a checking account which incorporates the following functionality:**
   - **Checking accounts can use an overdraft facility, so the new class will need instance variables for the overdraft limit and the amount overdrawn.**
   - **The overdraft should not be used unless the current balance is insufficient to cover a withdrawl.**
   - **Checking account holders may be charged for withdrawls or transfers, so the new class will also need a new instance variable for the number of transactions that have been made.**
   - **A new constructor will be required that accepts the account name, id, initial balance and overdraft limit. This constructor should use the super() facility to initialize the account id, name and balance.**

     This constructor should set the amount overdrawn to 0 and the transaction count to 0.
   - **Define another (overloaded) constructor which only accepts the account name, id and balance for the new CAccount (for customers who do not wish to use the overdraft facility yet).**

This constructor should set the overdraft limit to 0, the amount overdrawn to 0 and the transaction count to 0.

- **Decide whether accessors an/or mutators will be required for the three new instance variables.**

- **Define a new method getAvailableFunds() which returns the maximum amount the account holder can withdraw (including the overdraft facility) based on the following formula:**

  Total funds available = current balance + (overdraft limit - overdraft amount)

  Question: how do we access the current balance from the CAccount subclass?

- **The withdraw() method should be overridden incorporating the new overdraft facility into the withdrawl process as follows:**
  - **if the withdrawl amount is greater than the total funds available**
    - **Reject transaction**
  - **else**
    - **if the current balance is 0**
      - **Add withdrawl amount to amount overdrawn**
    - **else**
      - **subtract current balance from the withdrawl amount**
      - **withdraw the entire current balance**

        (Hint: super.withdraw() and the balance accessor might be useful here)
      - **add the remaining withdrawl amount to the amount overdrawn.**
    - **Increment the transaction count by 1**

- **The deposit method should be overridden to incorporate the new overdraft facility into the deposit process as follows:**
  - **if overdraft amount is 0**
    - **Add deposit amount to current balance**
  - **else if the deposit amount is less than or equal to the amount overdrawn**
    - **subtract deposit amount from amount overdrawn**
  - **else**
    - **subtract amount overdrawn from deposit amount**
    - **add remaining deposit amount to current balance**

- **Define a new method which deducts charges from the account as follows:**
  - **A fixed charge of $1.00 is applied for each additional transaction after the first 10.**
  - **An administration fee of 1% of the overdraft limit for the account is also charged each month.**
  - **The charges are to be deducted as follows:**
    - **if the charges for the month are less than or equal to the total funds available**
      - **process the charge deduction as a normal withdrawl.**
    - **else**
      - **subtract the total funds available from the charges amount**
      - **withdraw the total funds available**

        Hint: super.withdraw() and the getAvailableFunds() method might be useful here
      - **add the remaining charges to the overdraft amount.**
    - **reset the transaction count to 0**

Tutorial 9 – Polymorphism and other OO concepts

Review Questions:

1. What does the term polymorphism refer to?
2. What is dynamic binding and how does it work?
3. How can typecasting be used with object types?
4. What is the instanceof operator used for?
5. What does visibility refer to and how does it affect inheritance/accessibility between classes?
6. How does visibility affect the overriding and overloading of methods?
7. How do static class variables and methods differ from instance variables and methods?
8. What effect does the final modifier have on variables, method and classes?

Tutorial Exercises:

1. **The transfer () method of the superclass Account calls both deposit () and withdraw() methods as shown below.**

   Assume that withdraw() and deposit() methods are overridden in the subclasses SAccount and CAccount.

```
public boolean transfer(Account account, double amount)
{
   if (withdraw(amount))
   {
      account.deposit(balance);
      return true;
   }
   else
      return false;
}
```

   Which withdraw() and deposit() methods will be called as a result of the statements below ?

   SAccount acc1 = new SAccount("s12345","Clinton",1000);
   CAccount acc2= new CAccount("c12345","Bush",2000);

   a. **acc1.transfer(acc2,100);**

      withdraw() of class _____

      deposit() of class _____

   b. **acc2.transfer(acc1,150);**

      withdraw() of class _____

      deposit() of class _____

2. **Given that Salesman and Clerk are a subclasses of Employee and that all classes have default constructors what will be the result of the following?**

   (compilation error, run-time error, no problem)

   a. **Employee e = new Salesman();**
      Salesman s = (Salesman) e;

   b. **Employee e = new Salesman();**
      Salesman s = e;

   c. **Employee e1 = new Salesman();**
      Employee e2 = e1;

   d. **Employee e = new Salesman();**
      Clerk c = (Clerk) e;

   e. **Clerk c = new Salesman();**
      Salesman s = (Salesman) c;

3. **When is it necessary to cast a reference? When is it safe to do so?**

   a. **when a subclass reference is assigned its superclass reference**
   b. **when a superclass reference is assigned its subclass reference**

4. **Which of these conditions return true ?**

   Rectangle r = new Rectangle(5,10,15,25);

   a. **if (r instanceof Rectangle) …**
   b. **if (r instance of Point) …**
   c. **if (r instanceof Object) …**
   d. **if (r instance of RectangularShape) …**

5. **Suggest a scheme to get the number of objects created for particular class A.**

   Hint: use static (class) variables and methods.

6. **How can we prevent a subclass method overriding a superclass method ?**

   How can we prevent a class being inherited ?

7. **What kind of instance variables can be accessed in the same package ?**

8. **Superclass method protected void add(); can be overridden in the subclass by**

   a. **public void add();**
   b. **protected void add();**
   c. **void add();**
   d. **private void add();**

9. **Given that** `x.y();` **is a valid statement in Java, x is ____ y is _____**

   a. **x is an object; y a method of that class**
   b. **x is a class; y is a method**
   c. **x is either a class or object; y is either a method of that class or its superclasses**

COSC 2231 - Programming 1

Tutorial 10 – Abstract classes, interfaces and arrays

Review Questions:

1. What are abstract classes and when should you use them?
2. What restrictions are placed on abstract classes and their subclasses?
3. How do you implement an abstract class – is it any different from a "normal" class?
4. What is an interface in java and when should you use one?
5. What restrictions does an interface force upon the classes that implement it?
6. What relationship is there between abstract classes/interfaces and polymorphism?
7. How is a 2-dimensional array structured and how do you acess the elements within it?
8. What is a "foreach" loop and how does it differ from a for loop?

Tutorial Exercises:

1. **Given an array of 100 int elements named marks with n (<100) elements set as in:**

   int[] marks = new int[100];

   marks[0] = ..

   marks[1] = ..

   ..

   Write the code to

   a. **print all the elements** (using both a for loop and a foreach loop)

   b. **print all the elements with even index (including 0)**

   c. **print the largest element** (using a foreach loop)

   d. **insert a new element into an array in order, shifting all elements larger than the new element along to make room for its insertion**

2. **An organization has three types of employees, salesmen, hourly-rate employees and fixed income earners. For hourly rated employees salary is computed on hourly rate and the number of hours. If the number of hours exceed 150 (a month) they are paid a premium rate (1.5 times). For salesmen monthly income is 10% of the sales for that month. Fixed income earners have a fixed component and a productivity bonus.**

   a. **Given that that you are required to keep all employee objects in an array of superclass (Employee) references and compute the salary in a polymorphic way, suggest one abstract method for the superclass. Provide a possible implementation for this method in all its subclasses (state your assumptions).**

   b. **Given that these classes have constructors as shown below write a simple loop to show how the salaries can be computed in a polymorphic way.**

```
Salesman       Salesman(String name, String ID, double sales)
HourlyRated    Worker(String name, String ID, double rate, double hours)
FixedIncome    FixedIncome(String name, String ID, double amt,
                           double bonus)

Salesman       Bill Jones    d12345    120000.0
Hourly rated   Timothy       d23668    12.0      180
Salesman       Mark Cheng    f13456    60000.0
```

```
Fixed income   John Kennedy  f23456    3000.0    250.0
```

3. **Assume that you are hired as a consultant during your vacation to help implement the GST computation for a small computer shop. Currently there are two classes one for *products* and one for *services*. These classes compute the charges taking into account various discounts offered for each transaction. However they lack the feature to compute GST.**

   Assume that Product class has a method getTotalPrice() and Service class has methods getTransportCost() and getServiceCharge(). (For services there may be an additional transport charge depending on the distance – but assume that transport charges do not attract GST).

   You are required to derive new classes to help compute GST due for both products and services. Instances of these classes will be kept in a common array for the purpose of computing the total GST.

   Suggest an appropriate interface to allow computation of GST. Suggest how an array of this interface can be used for computing the total GST for all products and services.

COSC 2231 - Programming 1

Tutorial 11 – Exceptions and files

Review Questions:

1. What is an Exception and what does it represent?
2. How do you handle an exception in a program where it occurs?
3. What is exception propagation and how does it work?
4. How do you create and use custom designed exception types?
5. How do you open text files for reading and writing?
6. How do you read different types of data from a text file (ie. strings, numbers, etc)?
7. How do you write different types of data to a text file?
8. What do you do with text files after you have finished reading from or writing to them?

Tutorial Exercises (Exceptions):

1. **What is the output of the following code?**

```
int theValue = 30;
try
{
   System.out.println("Block entered");
   if (theValue > 10)
      throw new Exception ("Value exceeded");
   System.out.println("Leaving block");
}
catch (Exception e)
{
   System.out.println("Exception occurred: " + e.getMessage());
}
System.out.println("After catch block");
```

What is the output if we change the top line to int theValue = 0;

2. **Consider the following code:**

```
try
{
   statement1;
   statement2;
   statement3;
}
catch (Exception1 e1)
{
     ……
}
catch (Exception2 e2)
{
     ……
};
finally {
  statement4;
}
statement5;
```

If statement2 causes an exception, decide whether the following are true or false:

a. **statement3 is executed**
b. **if the exception isn't caught, statement5 is executed**
c. **if the exception is caught in the catch clause, statement5 is executed**
d. **statement4 is executed**

3. **Assume SubException extends Exception. Consider the following code:**

```
try
{
   statement1;
   statement2;
}
catch (Exception e)
{
    ……
}
catch (SubException se)
{
   ……
};
```

What happens if any of the statements throws a SubException?

Is there something wrong with this code?

If the scenario changes so that SubException1 and SubException2 extend Exception (but not each other) and the code is:

```
try
{  statement1;
   statement2;
}
catch (SubException1 se1)
{   ……
}
catch (SubException2 se2)
{   ……
};
```

What happens if any of the statements throws a SubException1 or SubException2?

Is there something wrong with this code?

What happens if any of the statements throws an Exception?

Tutorial Exercises (Files):

1. **Supposing you are given an Account array storing a number of Account (and its subclass) objects.**
   Discuss how such objects can be written to a text file and read back later.

2. **Write a program that will count the number of characters in a specified text file including blanks.**

3. **Distinguish between binary files and text files. If speed of writing and reading as well as accuracy is important which type of file will you choose ?**

4. **If we are to store large amount of fixed length records with direct access what kind of file would you choose ?**

   [ ] a text file

   [ ] a binary file

   Supposing we are required to access record 2001 directly where each record has a fixed size of 1000 write a statement which will set the file position correcly.

   r.seek(_____);